

# LAB CYCLE 1

## 1. Program to Print all non-Prime Numbers in an Interval

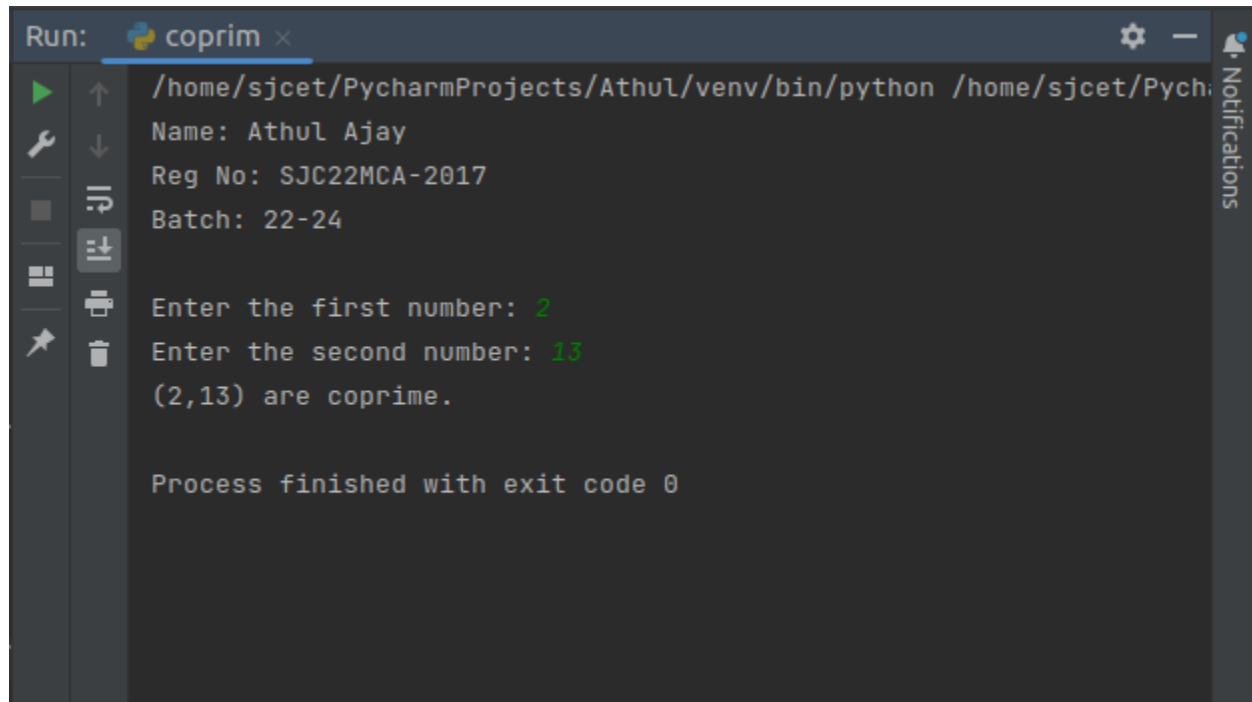
```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-017")
print("Batch: 22-24")
print()
lower = int(input("Enter the first number: "))
upper = int(input("Enter the second number: "))

print("Non Prime numbers between", lower, "and", upper, "are:")

for num in range(lower, upper + 1):

    if num > 1:
        for i in range(2, int(num ** 0.5)+1):
            if (num % i) == 0:
                break
        else:
            continue
    print(num)
```

## OUTPUT:



The screenshot shows a PyCharm Run console window titled "Run: coprim x". The console output is as follows:

```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/PycharmProjects/Athul/coprim.py  
Name: Athul Ajay  
Reg No: SJC22MCA-2017  
Batch: 22-24  
  
Enter the first number: 2  
Enter the second number: 13  
(2,13) are coprime.  
  
Process finished with exit code 0
```

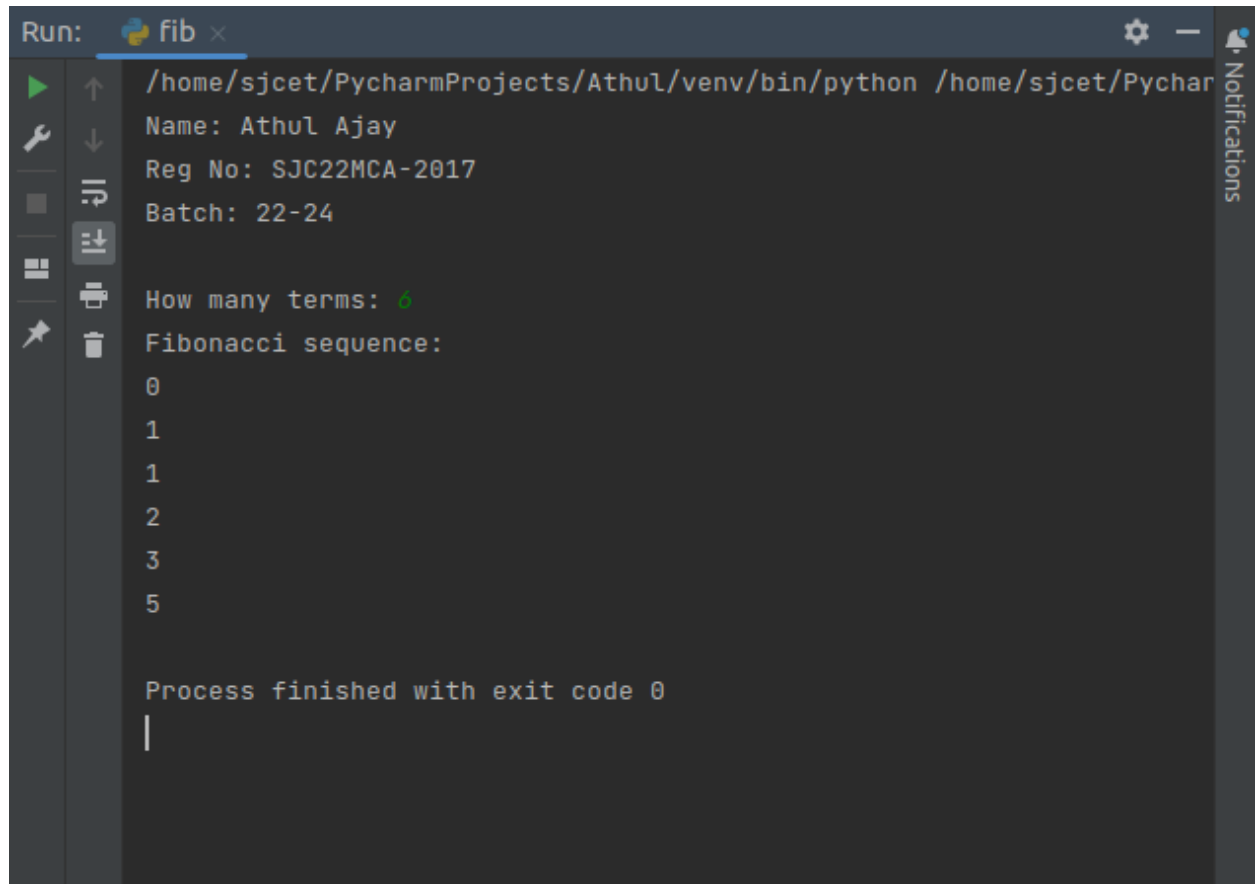
The input numbers 2 and 13 are highlighted in green in the original image. The console window includes a toolbar on the left with icons for running, debugging, and other actions, and a "Notifications" panel on the right.

## 2. Program to print the first N Fibonacci numbers.

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-017")
print("Batch: 22-24")
print()
nterms = int(input("How many terms: "))
n1 = 0
n2 = 1
count = 0

if(nterms <= 0):
    print("Please enter a positive integer!")
elif(nterms == 1):
    print("Fibonacci sequence upto",nterms,":")
    print(n1)
else:
    print("Fibonacci sequence:")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        n1 = n2
        n2 = nth
        count += 1
```

## OUTPUT:



The image shows a PyCharm Run window for a file named 'fib'. The window has a dark theme and a sidebar on the left with icons for Run, Debug, and other actions. The main area displays the output of the script, which includes the file path, user information, and the Fibonacci sequence for 6 terms. The process finished with exit code 0.

```
Run: fib x
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pychar
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24
How many terms: 6
Fibonacci sequence:
0
1
1
2
3
5
Process finished with exit code 0
```

3. Given sides of a triangle, write a program to check whether a given triangle is an isosceles, equilateral or scalene.

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-017")
print("Batch: 22-24")
print()
print("Enter a sides of the triangle: ")
a = int(input("a: "))
b = int(input("b: "))
c = int(input("c: "))

if a == b == c:
    print("Equilateral triangle")
elif a==b or b==c or a==c:
    print("Isosceles triangle")
else:
    print("Scalene triangle")
```

**OUTPUT:**



```
Run: tri x
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/PycharmProjects/Athul/main.py
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Enter a sides of the triangle:
a: 4
b: 5
c: 4
Isosceles triangle

Process finished with exit code 0
```

#### 4. Program to check whether given pair of number is coprime

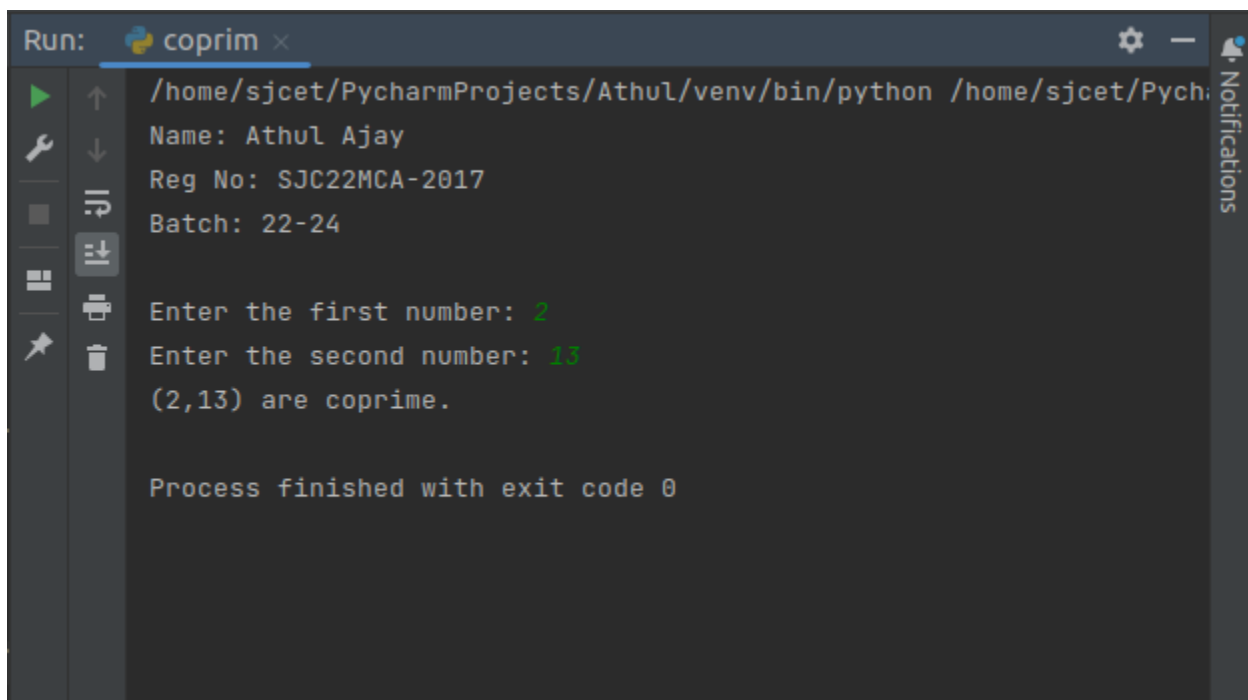
```
def calculate_gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def are_coprime(a, b):
    gcd = calculate_gcd(a, b)
    return gcd == 1

# Input two numbers from the user
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-017")
print("Batch: 22-24")
print()
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

if are_coprime(num1, num2):
    print(f"({num1},{num2}) are coprime.")
else:
    print(f"({num1},{num2}) are not coprime.")
```

#### OUTPUT:



The screenshot shows a PyCharm Run console window titled "Run: coprim x". The console output displays the program's execution: it prints the user's name "Name: Athul Ajay", registration number "Reg No: SJC22MCA-017", and batch "Batch: 22-24". It then prompts for two numbers, with the user entering "2" for the first and "13" for the second. The program outputs "(2,13) are coprime." and finally shows "Process finished with exit code 0". The console window includes standard PyCharm icons on the left and a "Notifications" panel on the right.

```
Run: coprim x
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pych
Name: Athul Ajay
Reg No: SJC22MCA-017
Batch: 22-24

Enter the first number: 2
Enter the second number: 13
(2,13) are coprime.

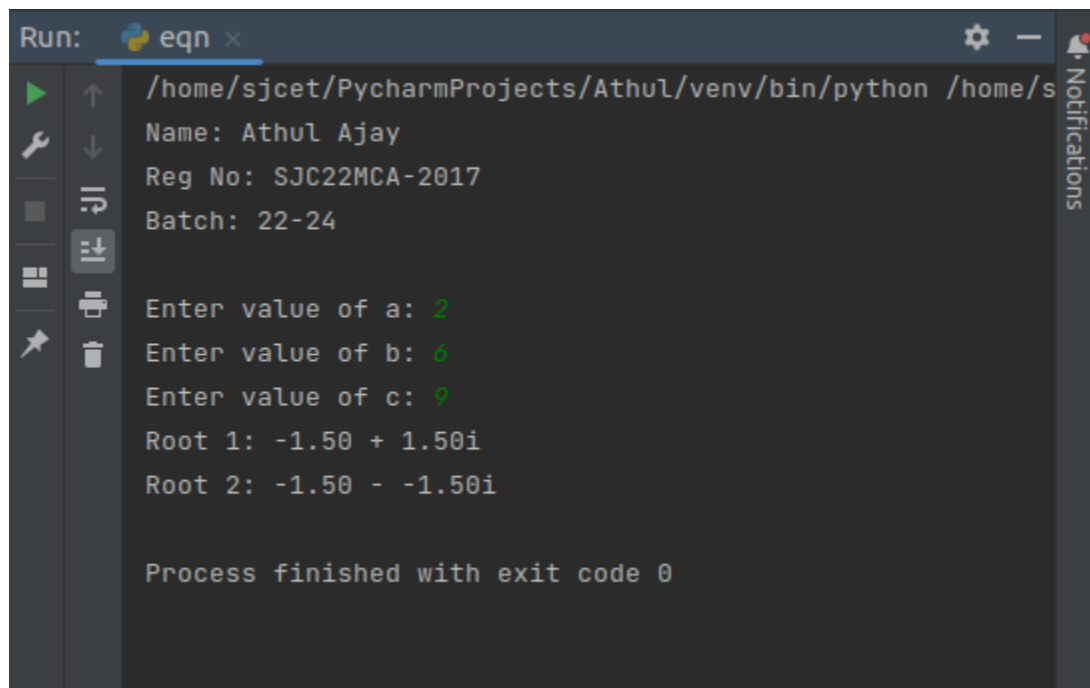
Process finished with exit code 0
```

## 5. Program to find the roots of a quadratic equation (rounded to 2 decimal places)

```
import math
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
a = float(input("Enter value of a: "))
b = float(input("Enter value of b: "))
c = float(input("Enter value of c: "))
discr = b**2 - 4*a*c

if discr > 0:
    root1 = (-b + math.sqrt(discr)) / (2*a)
    root2 = (-b - math.sqrt(discr)) / (2*a)
    print(f"Root 1: {round(root1, 2)}")
    print(f"Root 2: {round(root2, 2)}")
elif discr == 0:
    root = -b / (2*a)
    print(f"Root: {round(root, 2)}")
else:
    real_part = -b / (2*a)
    img_part = math.sqrt(-discr) / (2*a)
    root1 = complex(real_part, img_part)
    root2 = complex(real_part, -img_part)
    print(f"Root 1: {root1.real:.2f} + {root1.imag:.2f}i")
    print(f"Root 2: {root2.real:.2f} - {root2.imag:.2f}i")
```

## OUTPUT:

A screenshot of the PyCharm Run console window. The title bar shows 'Run: eqn' with a close button. On the left is a vertical toolbar with icons for play, step up, step down, run and debug, print, and a star. The main area displays the execution output in a monospaced font. It starts with the command path, followed by user input for name, registration number, and batch. Then it prompts for values of a, b, and c, which are entered as 2, 6, and 9 respectively. The program then calculates and displays two complex roots. The console ends with a message indicating the process finished successfully.

```
Run: eqn x
/home/sjcet/PycharmProjects/Athu1/venv/bin/python /home/s
Name: Athu1 Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Enter value of a: 2
Enter value of b: 6
Enter value of c: 9
Root 1: -1.50 + 1.50i
Root 2: -1.50 - -1.50i

Process finished with exit code 0
```



**6.**Program to check whether a given number is perfect number or not (sum of factors =number)

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
```

```
def is_perfect_number(num):
    if num <= 0:
        return False

    factors_sum = 1

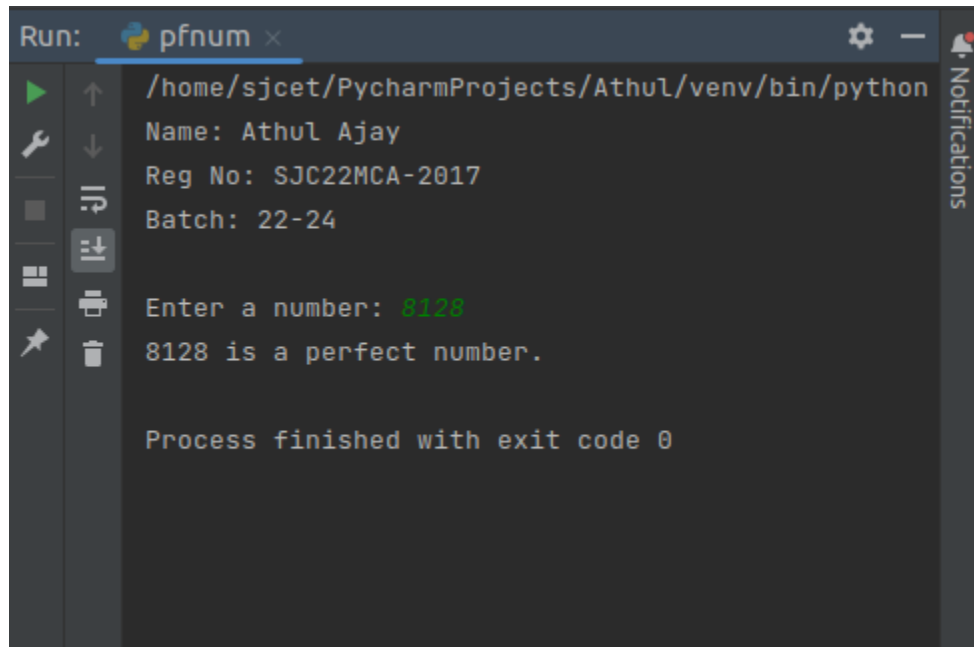
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            factors_sum += i
            if i != num // i:
                factors_sum += num // i

    return factors_sum == num

num = int(input("Enter a number: "))

if is_perfect_number(num):
    print(num, "is a perfect number.")
else:
    print(num, "is not a perfect number.")
```

## OUTPUT:



The image shows a PyCharm Run console window for a program named 'pfnum'. The console output displays the program's execution path, user input, and the final result. The user entered '8128' when prompted, and the program correctly identified it as a perfect number. The console also shows the program finished with an exit code of 0.

```
Run: pfnum x
/home/sjcet/PycharmProjects/Athul/venv/bin/python
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24
Enter a number: 8128
8128 is a perfect number.

Process finished with exit code 0
```

## 7. Program to display amstrong numbers upto 1000

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
lower = 1
upper = 1000

for num in range(lower, upper + 1):

    order = len(str(num))

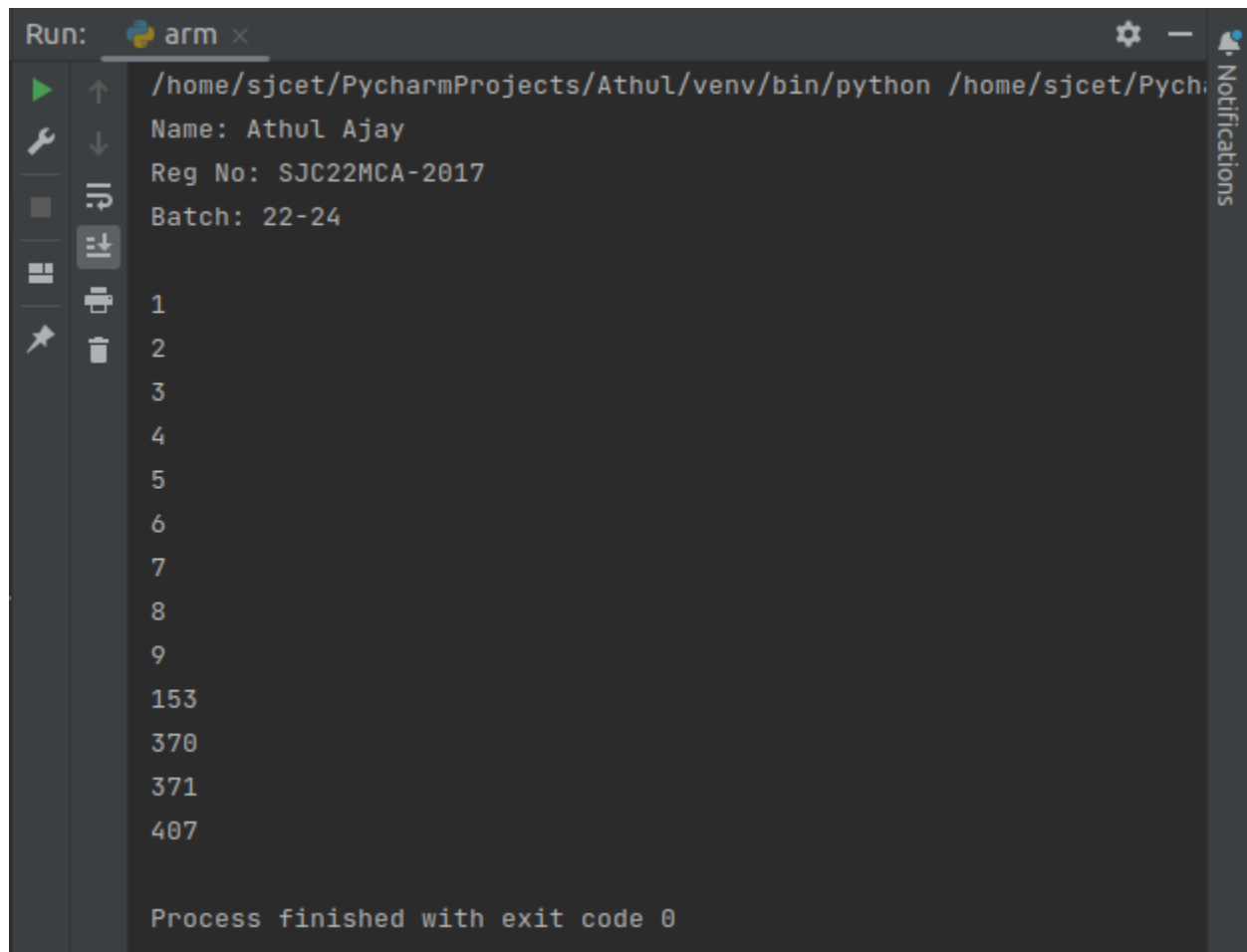
    sum = 0

    temp = num
    while temp > 0:
        digit = temp % 10
        sum += digit ** order
        temp //= 10

    if num == sum:

        print(num)
```

## OUTPUT:



The screenshot shows a PyCharm Run console window with a dark theme. The title bar reads "Run: arm x". The console output is as follows:

```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/PycharmProjects/Athul/main.py  
Name: Athul Ajay  
Reg No: SJC22MCA-2017  
Batch: 22-24  
  
1  
2  
3  
4  
5  
6  
7  
8  
9  
  
153  
370  
371  
407  
  
Process finished with exit code 0
```

The left sidebar contains standard IDE icons: Run, Debug, Run with Coverage, Run with Profiler, Run with Coverage and Profiler, and a star icon. The right sidebar shows a "Notifications" panel.

**8.Store and display the days of a week as a List, Tuple, Dictionary, Set.**  
Also demonstrate different ways to store values in each of them. Display its type also.

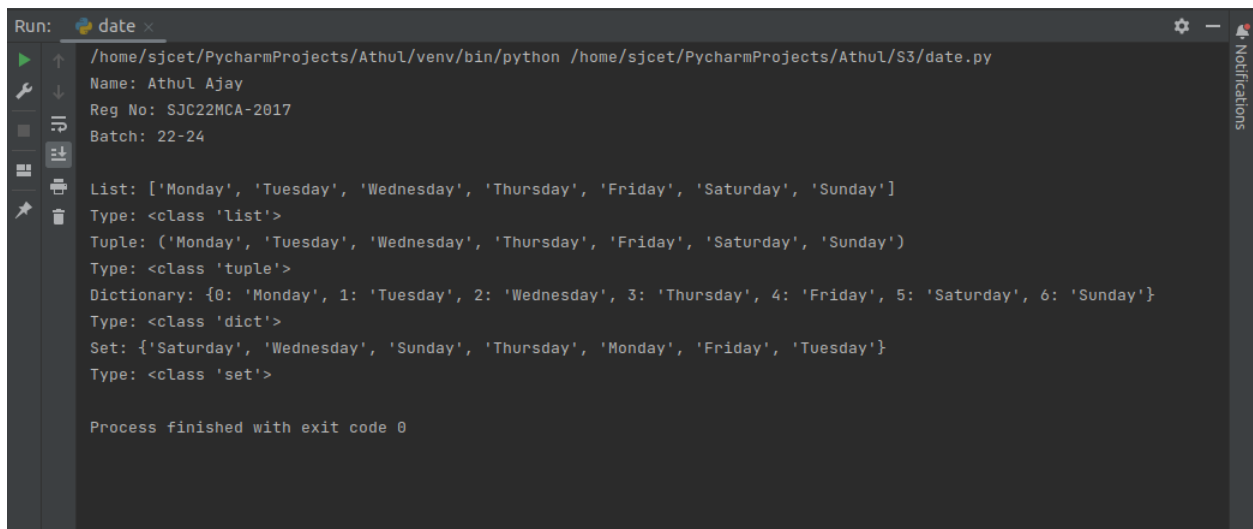
```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
days_list = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"]
print("List:", days_list)
print("Type:", type(days_list))

days_tuple = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday")
print("Tuple:", days_tuple)
print("Type:", type(days_tuple))

days_dict = {0: "Monday", 1: "Tuesday", 2: "Wednesday", 3: "Thursday", 4: "Friday", 5:
"Saturday", 6: "Sunday"}
print("Dictionary:", days_dict)
print("Type:", type(days_dict))

days_set = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"}
print("Set:", days_set)
print("Type:", type(days_set))
```

## OUTPUT:



```
Run: date x
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/PycharmProjects/Athul/S3/date.py
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

List: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
Type: <class 'list'>
Tuple: ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')
Type: <class 'tuple'>
Dictionary: {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Friday', 5: 'Saturday', 6: 'Sunday'}
Type: <class 'dict'>
Set: {'Saturday', 'Wednesday', 'Sunday', 'Thursday', 'Monday', 'Friday', 'Tuesday'}
Type: <class 'set'>

Process finished with exit code 0
```

### 9. Write a program to add elements of given 2 lists

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
def add_lists(list1, list2):

    if len(list1) != len(list2):
        return None

    result = []

    for i in range(len(list1)):
        result.append(list1[i] + list2[i])

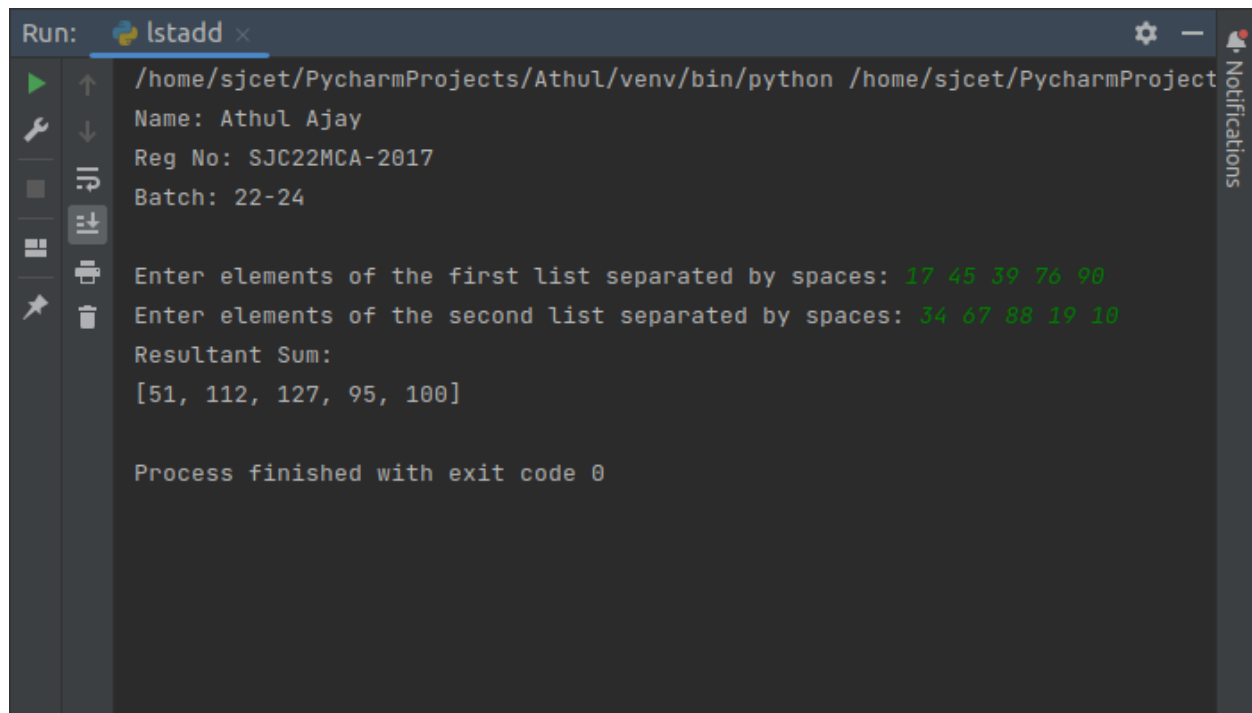
    return result

list1_str = input("Enter elements of the first list separated by spaces: ")
list2_str = input("Enter elements of the second list separated by spaces: ")

list1 = [int(x) for x in list1_str.split()]
list2 = [int(x) for x in list2_str.split()]

if len(list1) != len(list2):
    print("Lists are of different lengths!!")
else:
    result_list = add_lists(list1, list2)
    if result_list is not None:
        print("Resultant Sum:")
        print(result_list)
```

## OUTPUT:



The screenshot shows a PyCharm Run console window titled 'Run: lstadd'. The console output displays the program's execution, including user input for name, registration number, batch, and two lists of numbers. The resultant sum is calculated and displayed as a list. The process finished with exit code 0.

```
Run: lstadd x
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/PycharmProject
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24
Enter elements of the first list separated by spaces: 17 45 39 76 90
Enter elements of the second list separated by spaces: 34 67 88 19 10
Resultant Sum:
[51, 112, 127, 95, 100]

Process finished with exit code 0
```

**10.**Write a program to find the sum of 2 matrices using nested List.

```
print("Name: Athul Ajay")
```

```
print("Reg No: SJC22MCA-2017")
```

```
print("Batch: 22-24")
```

```
print()
```

```
def add_matrices(mat1, mat2):
```

```
    rows = len(mat1)
```

```
    cols = len(mat1[0])
```

```
    result = [[0 for _ in range(cols)] for _ in range(rows)]
```

```
    for i in range(rows):
```

```
        for j in range(cols):
```

```
            result[i][j] = mat1[i][j] + mat2[i][j]
```

```
    return result
```

```
rows = int(input("Enter the number of rows: "))
```

```
cols = int(input("Enter the number of columns: "))
```

```
matrix1 = []
```

```
matrix2 = []
```

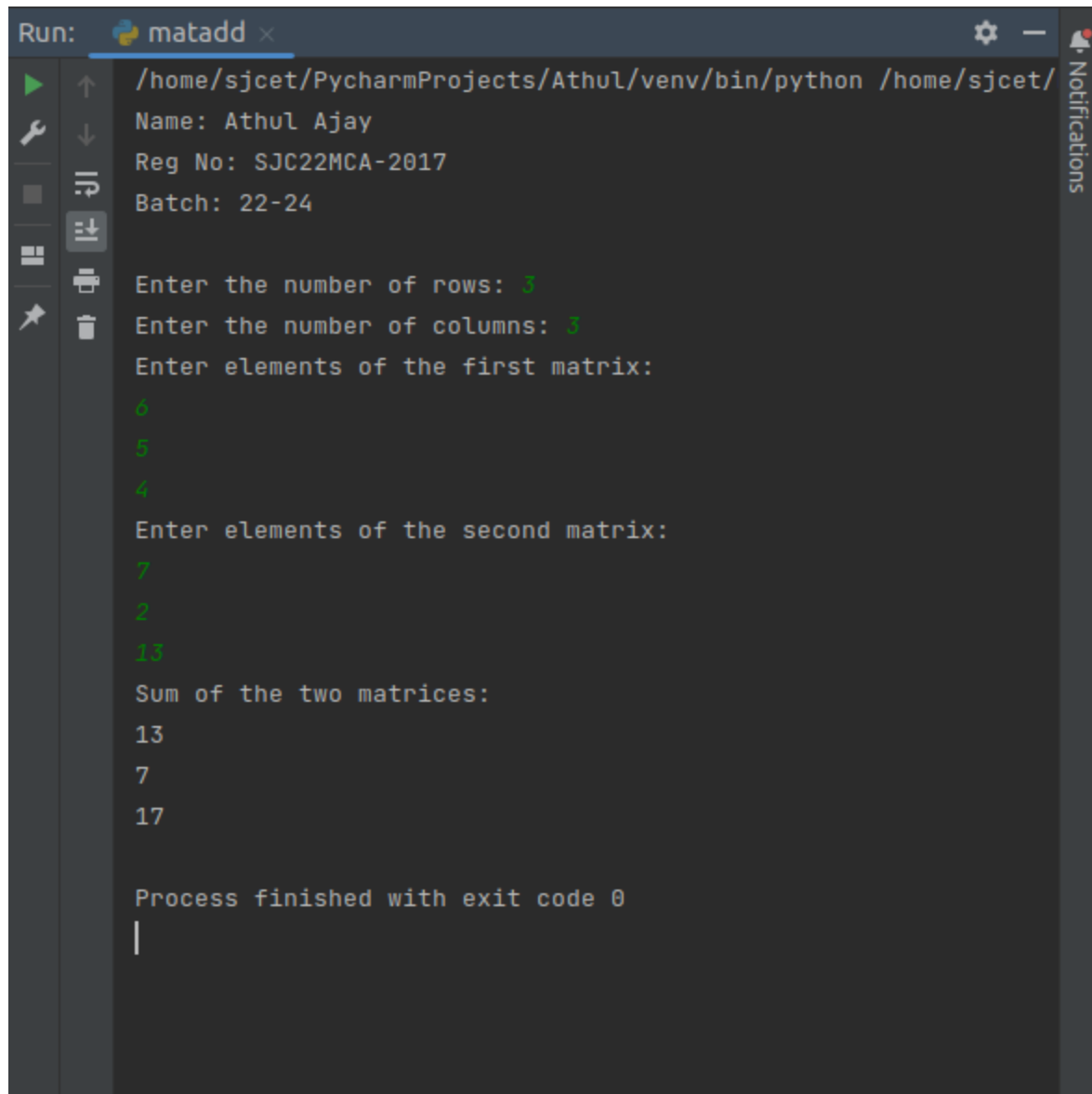


```
print("Enter elements of the first matrix:")
for i in range(rows):
    row = [int(x) for x in input().split()]
    matrix1.append(row)

print("Enter elements of the second matrix:")
for i in range(rows):
    row = [int(x) for x in input().split()]
    matrix2.append(row)

if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]):
    print("Matrices have different dimensions. Cannot perform addition.")
else:
    result_matrix = add_matrices(matrix1, matrix2)
    print("Sum of the two matrices:")
    for row in result_matrix:
        print(" ".join(map(str, row)))
```

## OUTPUT:



The screenshot shows a PyCharm Run window titled 'Run: matadd x'. The terminal output is as follows:

```
/home/sjcet/PycharmProjects/Athu1/venv/bin/python /home/sjcet/  
Name: Athu1 Ajay  
Reg No: SJC22MCA-2017  
Batch: 22-24  
  
Enter the number of rows: 3  
Enter the number of columns: 3  
Enter elements of the first matrix:  
6  
5  
4  
Enter elements of the second matrix:  
7  
2  
13  
Sum of the two matrices:  
13  
7  
17  
  
Process finished with exit code 0  
|
```

The input values for the first matrix are 6, 5, and 4. The input values for the second matrix are 7, 2, and 13. The resulting sum matrix contains the values 13, 7, and 17.

**11.**Write a program to perform bubble sort on a given set of elements.

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
def bubble_sort(arr):
    n = len(arr)

    for i in range(n):
        swapped = False

        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True

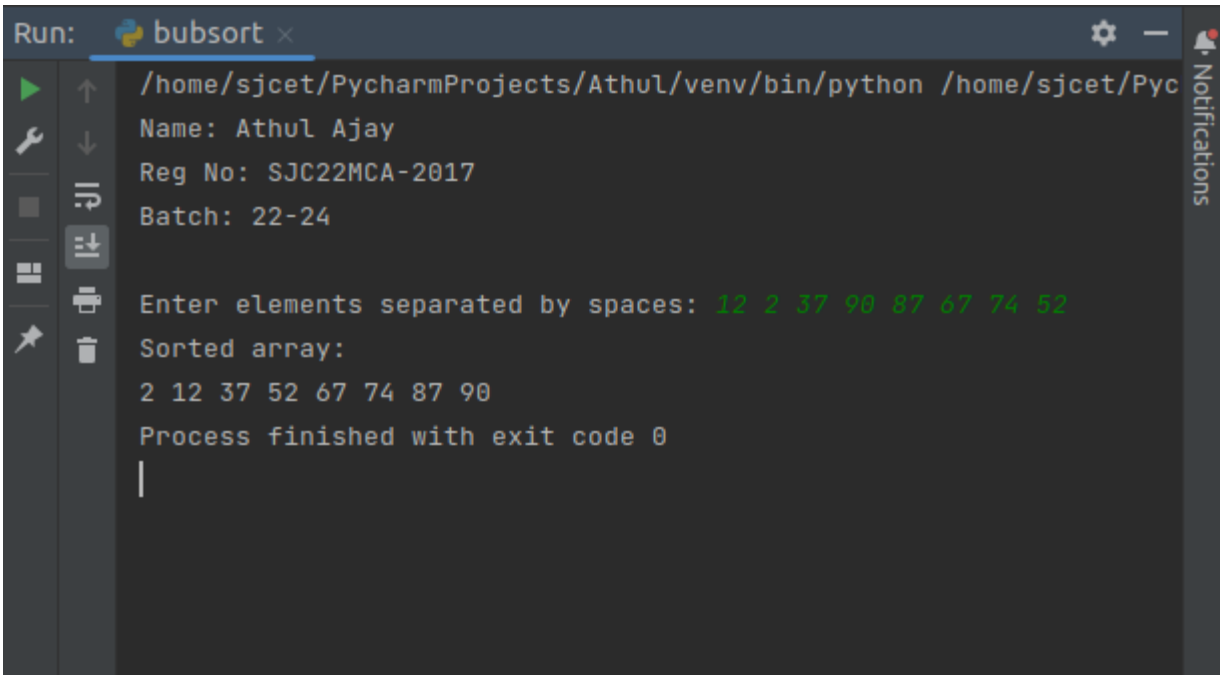
        if not swapped:
            break

str = input("Enter elements separated by spaces: ")
elements = [int(x) for x in str.split()]

bubble_sort(elements)

print("Sorted array:")
for element in elements:
    print(element, end=" ")
```

## OUTPUT:



```
Run: bubsort x
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pyc
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24
Enter elements separated by spaces: 12 2 37 90 87 67 74 52
Sorted array:
2 12 37 52 67 74 87 90
Process finished with exit code 0
|
```

## 12. Program to find the count of each vowel in a string (use dictionary)

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
def count_vowels(input_string):
    vowel_counts = {'a': 0, 'e': 0, 'i': 0, 'o': 0, 'u': 0,
                    'A': 0, 'E': 0, 'I': 0, 'O': 0, 'U': 0}

    for char in input_string:
        if char in vowel_counts:
            vowel_counts[char] += 1

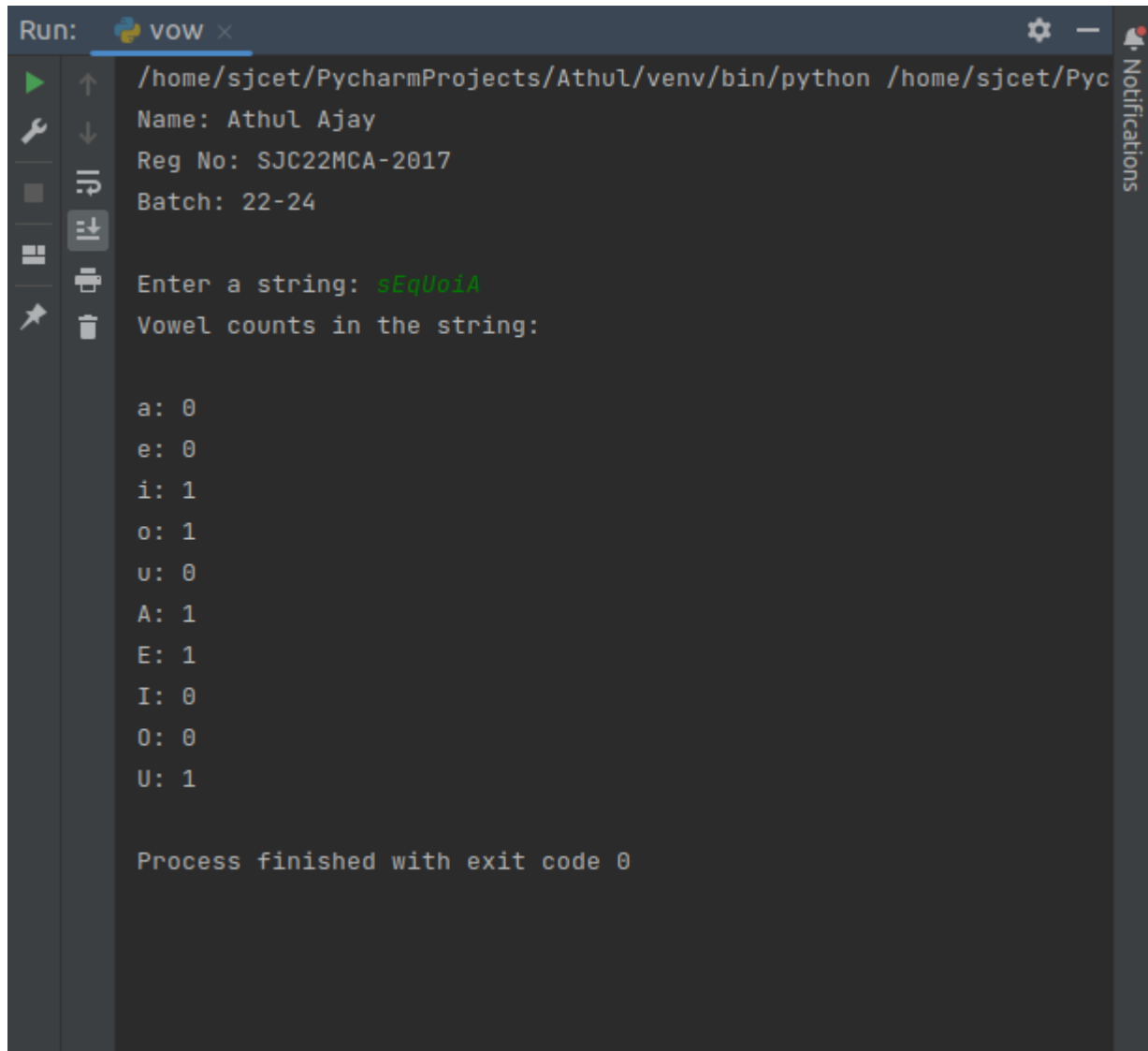
    return vowel_counts

input_string = input("Enter a string: ")

vowel_counts = count_vowels(input_string)

print("\Vowel counts in the string:")
print()
for vowel, count in vowel_counts.items():
    print(f"{vowel}: {count}")
```

## OUTPUT:



```
Run: vow x
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pyc
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24
Enter a string: sEqUoiA
Vowel counts in the string:

a: 0
e: 0
i: 1
o: 1
u: 0
A: 1
E: 1
I: 0
O: 0
U: 1

Process finished with exit code 0
```

**13.**Write a Python program that accept a positive number and subtract from this number the sum of its digits and so on. Continues this operation until the number is positive (eg:  $256 \rightarrow 2+5+6=13$ ,  $256-13=243$ ,  $243-9=232$ )

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
def sum_of_digits(number):

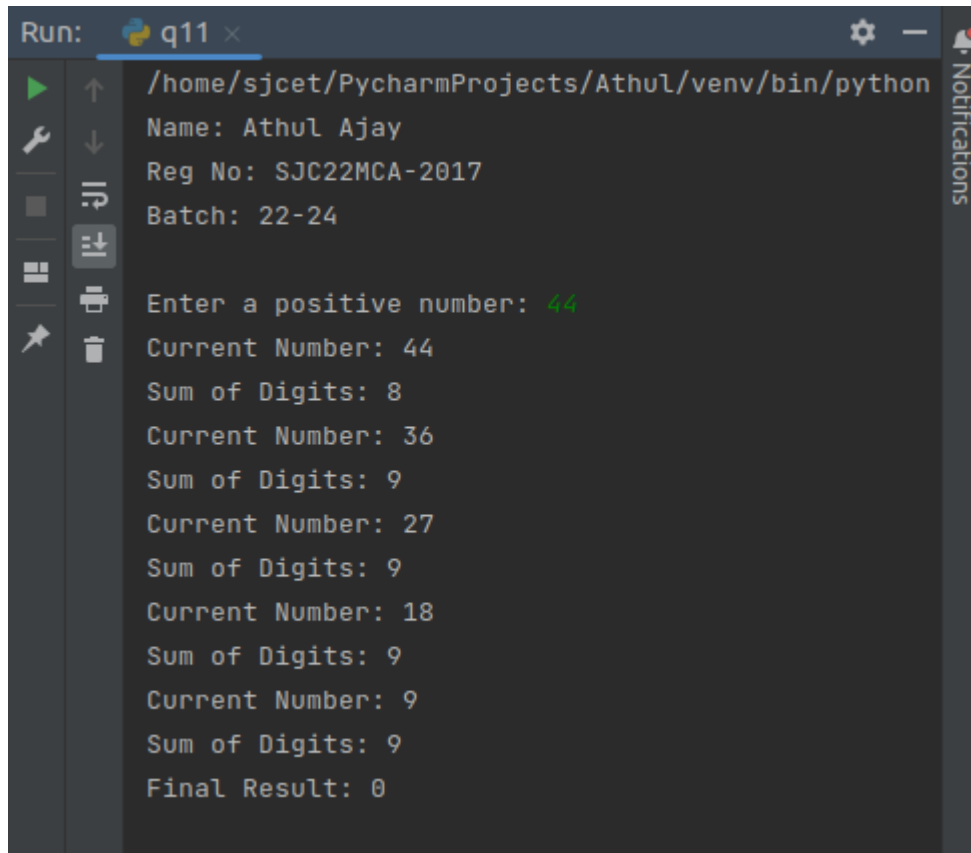
    digit_sum = 0
    while number > 0:
        digit_sum += number % 10
        number //= 10
    return digit_sum

def subtract_until_non_positive(number):
    while number > 0:
        print("Current Number:", number)
        digit_sum = sum_of_digits(number)
        print("Sum of Digits:", digit_sum)
        number -= digit_sum
    print("Final Result:", number)

num = int(input("Enter a positive number: "))

if num <= 0:
    print("Please enter a positive number.")
else:
    subtract_until_non_positive(num)
```

## OUTPUT:



```
Run: q11 x
/home/sjcet/PycharmProjects/Athul/venv/bin/python
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24
Enter a positive number: 44
Current Number: 44
Sum of Digits: 8
Current Number: 36
Sum of Digits: 9
Current Number: 27
Sum of Digits: 9
Current Number: 18
Sum of Digits: 9
Current Number: 9
Sum of Digits: 9
Final Result: 0
```



**14.**Write a Python program that accepts a 10-digit mobile number, and find the digits which are absent in a given mobile number

```
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
def find_absent_digits(mobile_number):
    all_digits = set("0123456789")

    mobile_digits = set(mobile_number)

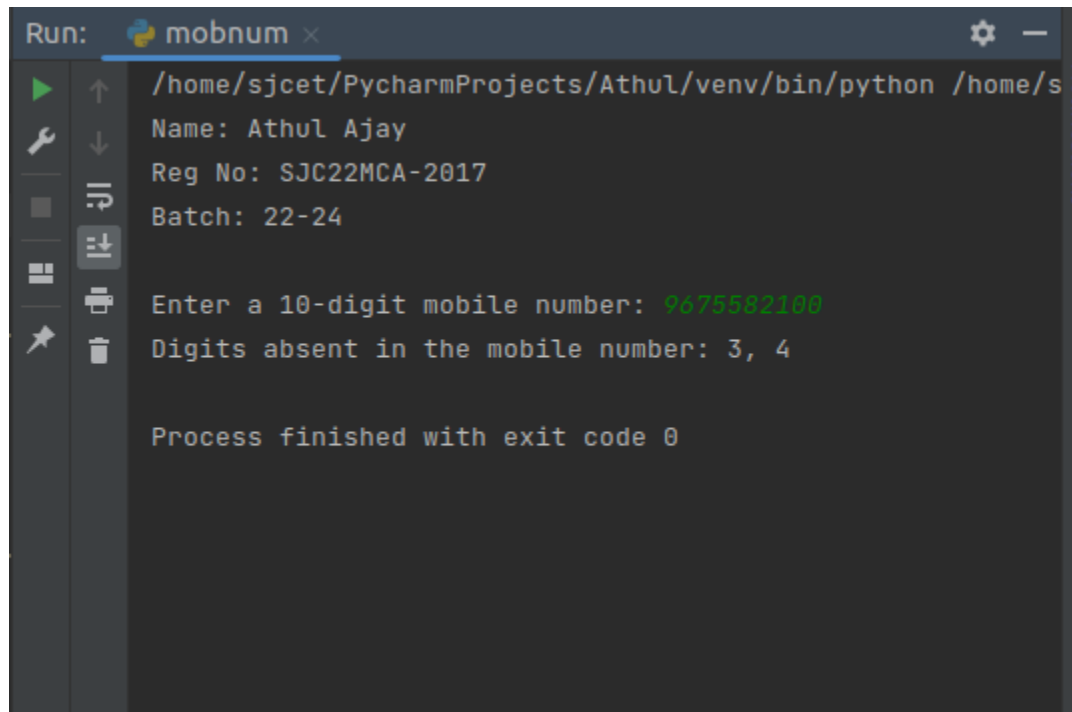
    absent_digits = all_digits - mobile_digits

    return sorted(list(absent_digits))

mobile_number = input("Enter a 10-digit mobile number: ")

if len(mobile_number) == 10 and mobile_number.isdigit():
    absent_digits = find_absent_digits(mobile_number)
    if absent_digits:
        print("Digits absent in the mobile number:", " ", ".join(absent_digits))
    else:
        print("All digits are present in the mobile number.")
else:
    print("Invalid input. Please enter a valid 10-digit mobile number.")
```

## OUTPUT:



```
Run: mobnum x
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/s
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Enter a 10-digit mobile number: 9675582100
Digits absent in the mobile number: 3, 4

Process finished with exit code 0
```